

**HONVÉLT**

SZOFTVER

A BASIC JELKÉSZLETE:

- számok: 0 1 2 3 4 5 6 7 8 9
- 31 db magyar kis- és nagybetű
- műveleti jelek: + - / \*
- relációk: < > =
- egyéb jelek: ! " # \$ % & ' ( ) , . : ; ?

A BASIC által használt speciális karakterek a következők:

- CR CARRIAGE RETURN. A begépelte sort zárja le.
- CLS Clear. Shift/CR-rel érhető el; a képernyőt törli.
- HOME HOME. Shift/space-vel érhető el; a kurzort a kép tetejére teszi.
- F1/F2 Az F1 és F2 funkciójú gombok egyidejű lenyomásával érhető el a BREAK, amely a BASIC program futását szakítja meg. Más gépeken általában a CONTROL/C. Csak a program futása közben érvényes.
- CURSOR Mozgató karakterek. A CURSOR segítségével a képernyő bármely pontja elérhető, ott javítás végezhető vagy a szöveg újraírható. A CR hatására mindig az a sor kerül bevételre, amelyikben a CURSOR éppen volt. Itt logikai sor értendő, nem fizikai!
- INS INSERT. A shift/→ segítségével érhető el, új karakter bevezetésére alkalmas. A CURSOR-tól jobbra eső szövegrészt eggyel jobbra lépteti, így a CURSOR helyén egy szabad karakter marad. Ismételt megnyomással újabb betűkhöz karakterek keletkeznek, amelyek helyébe új szöveg írható be.
- DEL DELETE. A shift/← segítségével érhető el, a egy karakter törlésére szolgál. A CURSOR-tól jobbra eső szövegrészt eggyel balra lépteti úgy, hogy a CURSOR által kijelölt karakter elvész. Ha a CURSOR sor végén van, az utolsó karaktert törli.

# A BASIC JELKÉSZLETE:

- számok: 0 1 2 3 4 5 6 7 8 9
- 31 db magyar kis- és nagybetű
- műveleti jelek: + - / \*
- relációk: < > =
- egyéb jelek: ! " # \$ % & ' ( ) , . : ; ?

## A BASIC által használt speciális karakterek a következők:

- CR CARRIAGE RETURN. A begépelte sort zárja le.
- CLS Clear. Shift/CR-rel érhető el; a képernyőt törli.
- HOME HOME. Shift/space-vel érhető el; a cursort a kép tetejére teszi.
- F1/F2 Az F1 és F2 funkciójú gombok egyidejű lenyomásával érhető el a BREAK, amely a BASIC program futását szakítja meg. Más gépeken általában a CONTROL/C. Csak a program futása közben érvényes.
- CURSOR Mozgató karakterek. A CURSOR segítségével a képernyő bármely pontja elérhető, ott javítás végezhető vagy a szöveg újraírható. A CR hatására mindig az a sor kerül bevételre, amelyikben a CURSOR éppen volt. Itt logikai sor értendő, nem fizikai!
- INS INSERT. A shift/→ segítségével érhető el, új karakter bevezetésére alkalmas. A CURSOR-tól jobbra eső szövegrészt eggyel jobbra lépteti, így a CURSOR helyén egy szabad karakter marad. Ismételt megnyomással újabb betűkhöz karakterek keletkeznek, amelyek helyébe új szöveg írható be.
- DEL DELETE. A shift/← segítségével érhető el, a egy karakter törlésére szolgál. A CURSOR-tól jobbra eső szövegrészt eggyel balra lépteti úgy, hogy a CURSOR által kijelölt karakter elvész. Ha a CURSOR sor végén van, az utolsó karaktert törli.



- BELL      BELL. Shift/↓ érhető el. Hatására a gép rövid hangjelzést ad. Stringbe beépítve így fűtty is printelhető.
- TAB      Tabulátor. Shift/↑ érhető el. Hatására a cursor új zóna elejére lép, és az útjában levő karaktereket törli.
- ALT      ALTER. Ez a gomb a nagybetűs és kisbetűs írásmód felcserélésére szolgál, csak a betűkre van hatásal.  
Ha a cursor teli négyzet alakjában villog, akkor a leütött gombok nagybetűt adnak, és a shift vált kisbetűre. Az ALT ezt úgy változtatja meg, hogy egy üres négyzet fog villogni, és alapesetben kisbetű, shift-tel pedig nagybetű lesz. Ujabb ALT visszavált az előbbi üzemmódra és így tovább.
- SPACE      Soremeléskénél, ha teli a kép és ezt a gombot nyomva tartjuk, egyenként lassan lépteti a sorokat. Ha shift-tel együtt nyomjuk meg, teljesen leáll a kiírás, egészen addig, míg újra shift/space-t nem nyomunk. Amíg áll a kép, addig is van lehetőség a sorokat egyenként léptetni a SPACE nyomogatásával. A shift/CR ebben az állapotban egy teljes új képet hoz be. Mindezeknek listázásnál és folyamatosan keletkező eredményeknél van szerepe.
- :
- ;
- #
- '
- Egy BASIC sorban tetszőleges számú utasítás lehet, ezek között kettőspont az elválasztó karakter.
- Lista elemeket választ el vagy a PRINT utasítás formátumát határozza meg.
- PRINT, INPUT és LIST utasításoknál OUTPUT, illetve INPUT eszközt jelöl ki.
- Feltételes listák elválasztó karaktere /lásd ON, IF/, vagy a BASIC sor hátralevő részének letiltására használható. Az ' mögött álló utasítások -az IF és az ON kivételével- sohasem hajtódnak végre. IF vagy ON után nem szerepelhet szövegkonstansban.

? Ekvivalens a PRINT kulcsszóval.

84.

" Stringkonstansok elejét és végét jelzi. Az " között lévő , . : ? karakterek elvesztik előbb felsorolt jelentésüket, ilyenkor ezek is hozzátartoznak a szövegkonstanshoz.

#### A SZÁMÁBRÁZOLÁSRÓL

A gép belső számábrázolása 4 bájtos lebegőpontos. Ennek megfelelően +1E38 és +1E-38 közé eső számokat tud ábrázolni, 1E-7 relatív hibakorláttal. A gép a számokat 6 jegyre kerekítve írja ki 999999 és 0.1 tartományban törtalakban, azon kívül pedig normál alakban. A mantissza és a karakterisztika + előjele, valamint a tizedespont előtti érvénytelen nullák elhagyhatók.

A duplapontos bővítésben a számok belső ábrázolása 16 jegyre nő és a gép 14 jegyet ír ki.

#### VÁLTOZÓK

A BASIC-ben használt változók azonosítója -típustól függetlenül- legfeljebb két betű lehet. A 30 nagybetűn kívül más karakter nem használható. Több betűből álló azonosítók esetében csak az első két betűt jegyzi meg és veszi figyelembe. A változók neve nem tartalmazhatja a kulcsszavak karaktersorozatát. A változók tartalmuk szerint REAL vagy STRING típusúak lehetnek. Ez utóbbinál az azonosító után áll.

Szervezésük szerint a változók lehetnek egyszerű változók, vektorok és mátrixok. A vektorok és mátrixok /tömbök/ azonosítója után zárójelben áll az egy-, ill. kéttagú kifejezés. Ugyanolyan típusú vektorok és mátrixok azonosítására nem használhatók azonos betűkombinációk. A legnagyobb tömbindex 255 lehet.

A BASIC-ban van öt speciális változó is, nevezetesen:

PI, INKEY, INKEY\$, CR, HM.

PI A  $\pi$  értékét adja: PI=3.14159

INKEY Ez a változó lehet REAL és STRING típusú is. Mind-  
INKEY\$ kettő az éppen lenyomott billentyűt adja: INKEY  
a lenyomott gomb ASCII kódját, az INKEY\$ pedig



magát a karaktert. Ha nincs lenyomva semmi, a5.  
INKEY=Ø és INKEY\$ üres string.

Az F1, F2 és SHIFT gombok lenyomását a gép külön nem veszi észre.

CR COLOR. A képernyőn PLOT utasítással megjeleníteni kívánt pont színét határozza meg. Ha CR=Ø fekete, ha CR=1 fehér a pont. A CR=2 invertálást jelent. CR-t csak akkor kell állítani, ha az éppen benne lévő szín nem megfelelő.

HM HIGHEST MEMORY. Az ide beírt decimális szám a BASIC számára felhasználható legmagasabb memóriahely címét jelenti. Ezzel a változóval gépkódú programrészletek és adatmezők számára rezerválható terület.

### KIFEJEZÉSEK

A kifejezésekre ugyanaz vonatkozik, mint a standard BASIC-ben. A prioritási elv és a balról-jobbra szabály érvényesül. Kifejezésekben használhatók az AND, OR, NOT műveletek és az összes relációk is. Az AND, OR, NOT logikai műveletek a számok egészrészének 2 bájtos fixpontos kettes komplementum alakjának bitjeire vonatkoznak. Pl.  
63 AND 16=16; 4 OR 2=6; NOT 1=-2

A relációknak szintén értékük van. Ha egy reláció hamis, akkor az értéke Ø, ha igaz értéke 1./A=2 esetén: (A=2)=1; (A=57)=Ø; (A>Ø)=1/

### STANDARD FÜGGVÉNYEK

Kevés kivételtől eltekintve a HOMELAB-BASIC az összes standard függvényt tartalmazza.

A függvények a következők:

ABS(X)	INT(X)	RND(X)	SGN(X)	SIN(X)
COS(X)	TAN(X)	ATN(X)	SQR(X)	EXP(X)
LOG(X)	FRE(X)	USR(X)	PEEK(X)	POINT(X,Y)

ABS(X)	X kifejezés abszolút értékét adja. ABS(1)=1; ABS(-2)=2
INT(X)	X kifejezés egész részét adja. INT(4.5)=4; INT(4)=4; INT(-3.2)=-4
SGN(X)	X kifejezés előjelét adja. Értéke +1, 0, -1 lehet. SGN(-3)=-1; SGN(0)=0; SGN(4.1)=1.
SIN(X)	X kifejezés sinusát adja; X értéke radiánban értendő.
COS(X)	X kifejezés cosinusát adja; X értéke radiánban értendő.
TAN(X)	X kifejezés tangensét adja; X értéke radiánban értendő.
ATN(X)	X kifejezés arcus tangensét adja. A függvény értéke radiánban adja a szöveget.
SQR(X)	X kifejezés négyzetgyökét adja. X értéke nem lehet negatív szám.
EXP(X)	e-t, a természetes logaritmus alapját; az X kifejezés által adott hatványra emeli.
LOG(X)	X kifejezés természetes alapú logaritmusát képezi.
RND(X)	Ha $X < 0$ , akkor ABS(X) a véletlenszám-generátor kezdőértéke lesz. Nagyságára nézve nincs kötés. Mivel RANDOMIZE utasítás nincs ebben a BASIC-ben, ez a tulajdonság használható véletlen sorozatok reprodukálására. Ha $X > 0$ , akkor $0 < \text{RND}(X) < X$ . X a véletlenszám intervalluma. Ha $X = 0$ , akkor RND(X) az a szám, amiből a következő véletlenszám generálódni fog.
PEEK(X)	Megadja az argumentumba írt kifejezés számértékének megfelelő című memóriahely tartalmát. Ha $X < 8192 / 255$ , akkor a PEEK nem memóriaterületet szólít meg, hanem az adott címmel IN gépkódú utasítást hajt végre.



Ily módon INPUT perifériák kezelhetők /több mint 8000 darab!/. a7.

Ha X nagyobb mint 65536, akkor X értékéből ez a szám levonódik, és a PEEK a második lap megfelelő címét olvassa.

**USR(X)** Gépikódú szubrutinhívást tesz lehetővé adatátvitellel. A szubrutincímet előzőleg POKE utasítással kell beállítani az USR-vektorba. A gépikódú szubrutin magas címbájtjának helye 16385, az alacsonyé pedig 16384. X egészrésze 2 bájtos fixpontos alakban a HL regiszterpárba kerül. RET gépikódú utasítás hatására a gépikódú rutin visszatér a BASIC-be, és USR(X) értéke a HL-ben lévő szám lesz. A CPU összes regisztere felhasználható IY kivételével.

**FRE(X)** Megadja a szabad memóriaterület nagyságát. X értéke közböns, de nem hiányozhat.

**POINT(X,Y)** Az X,Y koordinátájú képpont színét adja meg. Értéke 0, ha a pont fekete, 1, ha fehér. A 0, 0 pont a képernyő bal alsó sarka, és  $0 \leq X \leq 127$  vagy 63:  $0 \leq Y \leq 95$ .

#### STRINGEK, STRINGFÜGGVÉNYEK

Egy STRING hossza nem lehet nagyobb 255-nél és összesen 256-féle karakter fordulhat elő benne. Ezek mindegyike beírható a billentyűzetről is /a SHIFT, F1, F2 és egy jelgomb valamilyen kombinációjával/, és a CHR függvény segítségével is.

Van néhány karakter, amelynek speciális "nyomtatási képe" van. Ilyen, a karakterenként is kiadható: INS, DEL, BELL, TAB, HOME, CURSOR-mozgatások. Ezek a karakterek a billentyűzetről is beírhatók és beépíthetők a stringkonstansokba, mert az F1 vagy F2 hatására elvesztik speciális tulajdonságaikat, és csak a jelük látszik. Kinyomtatáskor természetesen mint speciális karakterek íródnak ki.



A HOMELAB BASIC-ben az alábbi stringfüggvények találhatók: <sup>a8.</sup>

ASC(X\$)

Megadja az X\$ első karakterének kódját. Ha ez a karakter az ASCII készlet eleme, akkor a kód egyben ASCII kód is.

CHR\$(I1, I2, ..., IN)

Azt a stringet adja meg, amely karaktereinek kódjai rendre: I k. I lehet speciális karakterek kódja is. ASC(CHR\$(I)) = I.

LFT\$(X\$, I)

X\$ első I karakteréből álló stringet adja meg.

RGH\$(X\$, I)

X\$ utolsó I karakteréből álló stringet adja meg.

MID\$(X\$, I, J)

X\$I-edik karakterénél kezdődő és J darab karakterből álló stringet adja meg.

LEN(X\$)

X\$ hosszának számértékét adja meg.

STR\$(X)

Azt a stringet adja meg, amelyik az X kifejezés értékének nyomtatási képe. Pl. X=3.1 STR\$(X)="\_3.1"

VAL(X\$)

Ha X egy számot ábrázol, vagy aritmetikai kifejezés stringképe, akkor VAL(X\$) ennek a számértékét adja.

Pl. A=3 és B=4 esetén VAL("A\*4+64\*B")= 268, vagy

VAL("-238.45") = 238.45 , VAL (" ") = 0

A stringekre értelmezve vannak a relációk és az összeadás is. Az összeadás egyszerű egymás után írást jelent. A\$=B\$, ha hosszuk egyenlő és elemeik rendre megegyeznek. A\$<B\$, ha B\$-ben előbb van nagyobb kódú karakter, mint A\$-ben, illetve A\$ kezdőstringje B\$-nek. /B\$ az elején tartalmazza A\$-et/.

Az alább felsorolt kulcsszavak a CONT kivételével utasítás-ként és parancsként egyaránt kiadhatók. /CONT -nem lehet utasítás./ Az utasítások zöme teljesen azonos a szabványos BASIC-kel.

Az utasítások és parancsok kulcsszókészlete:

BEEP	CALL	CONT	CUR	DATA	DIM
END	EXT	FOR	GOSUB	GOTO	IF
INPUT	LIST	LOAD	MON	NEXT	NEW
ON	PLOT	POKE	POP	PRINT	READ
REM	RESTORE	RETURN	RUN	SAVE	STEP
THEN	TO				

#### BEEP AS

Hangkeltő utasítás, elfűtyűli a mögöttes álló stringkifejezést. A stringbe az egyes karakterek különböző ritmusokat és hangmagasságoknak felelnek meg.

A karakterek jelentése a következő:

0	-	31	érvénytelen
32			zárókarakter, ez feltétlenül kell a string végén.
33	-	63	ritmust határoz meg. 33 a leggyorsabb sebesség, 63 a leglassabb sebesség.
64	-	255	hangmagasság. A 64 a legmagasabb, 255 a legmélyebb.

#### CALL X1, X2,...

Gépikódú szubrutin hívások az X1, X2 stb. /decimális/ címekre. A szubrutinhívások a címsorrendnek megfelelően egymás után következnek. A címek között vessző az elválasztó jel. Címnek nem csak szám, hanem aritmetikai kifejezés is írható. A szubrutin gépikódú RET utasítással tér vissza a BASIC-be. A CPU regiszterei IY kivételével felhasználhatók.

#### CONT

BRK-val megszakított program folytatását írja elő. Ez a folytatás csak akkor lehetséges, ha a BREAK üzenet óta még nem történt programsorbeírás, vagy bármilyen hiba üzenet. Ellenkező esetben CN ERROR keletkezik.



FOR X = A TO B STEP C: ..... :NEXT

Ciklusképző utasítás. X-a ciklus-változó, ami lehet vektor, vagy mátrix-elem is. A, B, C tetszőleges aritmetikai kifejezések. A ciklus-változó kezdőértéke A. B a ciklus végét adja, C pedig a növekmény. STEP C hiányozhat, ekkor a növekmény 1.

Ezután az ún. deklarativ rész után következik a ciklusmag, majd a NEXT cikluszáró utasítás. A ciklusváltozó értéke mindig a NEXT utasítás alatt módosul a növekménnyel. Ha  $C > 0$  a ciklus addig tart, míg  $X \leq B$ . Ha  $C < 0$  a ciklus  $X < B$ -ig jut. Ha a ciklus lejárt, a NEXT-et követő utasítás hajtódik végre.

Ciklusok tetszőleges mélységig egymásba skatulyázhatók.

Figyelem! A NEXT utasítás után nem szabad odairni a ciklusváltozót. Ezt más Basic-ek megengedik, vagy megkövetelik, de itt ez hibás!

Viszont egy NEXT-tel több ciklus is lezárható. A NEXT utasítás mögé írt minden vessző újabb ciklust zár le.

/pl. NEXT,, ekvivalens a NEXT : NEXT : NEXT-el./

FOR ciklus nélkül használt NEXT esetén Pp Error-t üzen.

GOTO X

Feltétlen vezérlésátadó utasítás. A program végrehajtása, az X sornál folytatódik. X lehet egy sorszám is, de lehet aritmetikai kifejezés is. Ezzel kiszámított GOTO képezhető. Ha X aritmetikai kifejezés, akkor nem kezdődhet számmal, vagy ha ez elkerülhetetlen, akkor zárójelbe kell tenni.

GOSUB X

Szubrutinhívás az X címre. X-re ugyanaz érvényes, mint a GOTO utasításnál. Lehetőség van egymás után több szubrutin meghívására is, ekkor a címeket vesszővel elválasztva kell felsorolni. A szubrutinok értelemszerűen a címek sorrendjében hajtódnak végre.

IF

Lásd az ON után!

FOR X = A TO B STEP C: ..... :NEXT

Ciklusképző utasítás. X-a ciklus-változó, ami lehet vektor, vagy mátrix-elem is. A, B, C tetszőleges aritmetikai kifejezések. A ciklus-változó kezdőértéke A. B a ciklus végét adja, C pedig a növekmény. STEP C hiányozhat, ekkor a növekmény 1.

Ezután az ún. deklarativ rész után következik a ciklusmag, majd a NEXT cikluszáró utasítás. A ciklusváltozó értéke mindig a NEXT utasítás alatt módosul a növekménnyel. Ha  $C > 0$  a ciklus addig tart, míg  $X \leq B$ . Ha  $C < 0$  a ciklus  $X < B$ -ig jut. Ha a ciklus lejárt, a NEXT-et követő utasítás hajtódik végre.

Ciklusok tetszőleges mélységig egymásba skatulyázhatók.

Figyelem! A NEXT utasítás után nem szabad odairni a ciklusváltozót. Ezt más Basic-ek megengedik, vagy megkövetelik, de itt ez hibás!

Viszont egy NEXT-tel több ciklus is lezárható. A NEXT utasítás mögé írt minden vessző újabb ciklust zár le.

/pl. NEXT,, ekvivalens a NEXT : NEXT : NEXT-el./

FOR ciklus nélkül használt NEXT esetén Pp Error-t üzen.

GOTO X

Feltétlen vezérlésátadó utasítás. A program végrehajtása, az X sornál folytatódik. X lehet egy sorszám is, de lehet aritmetikai kifejezés is. Ezzel kiszámított GOTO képezhető. Ha X aritmetikai kifejezés, akkor nem kezdődhet számmal, vagy ha ez elkerülhetetlen, akkor zárójelbe kell tenni.

GOSUB X

Szubrutinhívás az X címre. X-re ugyanaz érvényes, mint a GOTO utasításnál. Lehetőség van egymás után több szubrutin meghívására is, ekkor a címeket vesszővel elválasztva kell felsorolni. A szubrutinok értelemszerűen a címek sorrendjében hajtódnak végre.

IF

Lásd az ON után!



INPUT # X;A,B .....

al3.

Ez az INPUT-tal azonos módon működik, mindössze az a különbség, hogy az input-periféria nem a billentyűzet lesz. X szám vagy kifejezés azonosítja az input eszközt.

# 0-a billentyűzet. /Ez, mint az előbb is láttuk, el is hagyható./ További input perifériák kiszolgálására való rutinokat külön kell megadni. Ezt a gépkódú rész az INPUT # VEKTOR kapcsán ismerteti.

### LIST

Kilistázza a programot. Ha utána egy szám áll, akkor csak azt a sort írja ki. Ha két szám áll kötőjellel, akkor a két sorszám közti területet írja ki. Itt a kettő közül bármelyik szám hiányozhat, akkor értelemszerűen az adott sortól ill. az adott sorig listáz. Ha a LIST után # áll, akkor az előzővel azonos módon, de a printerre listáz.

### LOAD " NEV "

Kazettás megnóról programot olvas. A rekord nevét idéző jelek közé kell tenni. Ha a név hiányzik, akkor az első érvényes rekordot veszi be. Ha értelmes rekordot talál, kiírja a nevet. Ha a név nem egyezett, tovább keres. Ha nem volt név, vagy a név egyezett, beveszi a rekordot. LOAD után a CR-t még azelőtt meg kell nyomni, hogy a mag-nón a fűtő megzólal.

Ha a beolvasás hibás volt, de csak a program szövegében van hiba, akkor ERROR üzenettel tér vissza. Ha máshol volt a hiba, és a beolvasás teljesen használhatatlan, újra bejelentkezik a gép, és ekkor alapállapotba kerül a BASIC-is. A LOAD parancsot lehetőleg a kép tetején adjuk ki, mert az utolsó sorban a soremelés miatt hibás lesz a név kiírása.

### MON AS

Ez az utasítás a mögötte álló Stringkifejezésben megadott Monitor vagy Assembler parancsot hajtja végre. A Stringnek pontosan úgy kell kinéznie, mintha a Monitorban adták volna ki. A Monitor-parancs végrehajtása után visszatér a BASIC-be.

NEW

Alapállapotba hozza a BASIC-et. Ez azonos lesz a bekapcsolás utáni állapottal. Ha utána egy aritmetikai kifejezés is áll, akkor ez annak a memóriahelynek a decimális címe lesz, ahol a BASIC-program kezdődni fog.

ON X

Szokásos alakja ON X GOTO 10,20,30 MÁS GÉPEKEN!!!

Az eddig használt listáknál a , elválasztó jel minden listaelem figyelembevételét előírta. Pl. PRINT A,B A/X,Y/ GOSUB 10,20. Viszont ON utasításnál a listának csakis egyetlen /persze X-től függő/ eleme számít. Az ON utáni lista lehet tehát egy exkluzív lista, amely elemeinek elválasztására új jelet érdemes bevezetni.

Ez az aposztrof. Eszerint a helyes formátum a HOMELAB - BASIC-ben ON GOTO' 10'20'30 lesz. /Itt az első listaelem előtt is kell ' /. Az ON után tehát a GOTO utasítás "ki van emelve" és az utána következő exkluzív lista az argumentuma. Ennek analógiájára bármely utasítás -kivéve az értékadást- kiemelhető ON után.

Pl. ON X PRINT' A'B'C

X értéknek megfelelően A,B vagy C értéket nyomtatja ki.

Ha X különböző értékeire különböző utasításokat kell végrehajtani, tehát az utasítás nem emelhető ki, akkor az ON X után egy exkluzív utasítás-lista is állhat.

```
Pl. 10 ON X ' PRINT A : C = 3 ' PRINT D ' FOR J = 1
    TO 7 : U (J) = 0 : NEXT
    20 GOSUB 100 ...
```

Ezt a hagyományos módon a következő programrészlet oldja meg:

```
10 ON X GOTO 1000, 1100, 1200
20 GOSUB 100...
1000 PRINT A : C = 2 : GOTO 20
1100 PRINT D : GOTO 20
1200 FOR J = 1 TO 7 : U (J) = 0 : NEXT : GOTO 20
```



N E W

Alapállapotba hozza a BASIC-et. Ez azonos lesz a bekapcsolás utáni állapottal. Ha utána egy aritmetikai kifejezés is áll, akkor ez annak a memóriahelynek a decimális címe lesz, ahol a BASIC-program kezdődni fog.

ON X

Szokásos alakja ON X GOTO 10,20,30 MÁS GÉPEKEN!!!  
Az eddig használt listáknál a , elválasztó jel minden listaelem figyelembevételét előírta. Pl. PRINT A,B  
A/X,Y/ GOSUB 10,20. Viszont ON utasításnál a listának csakis egyetlen /persze X-től függő/ eleme számít. Az ON utáni lista lehet tehát egy exkluzív lista, amely elemeinek elválasztására új jelet érdemes bevezetni.  
Ez az aposztrof. Eszerint a helyes formátum a HOMELAB - BASIC-ben ON GOTO' 10'20'30 lesz. /Itt az első listaelem előtt is kell ' /. Az ON után tehát a GOTO utasítás "ki van emelve" és az utána következő exkluzív lista az argumentuma. Ennek analógiájára bármely utasítás -kivéve az értékadást- kiemelhető ON után.

Pl. ON X PRINT' A'B'C

X értéknek megfelelően A,B vagy C értéket nyomtatja ki.

Ha X különböző értékeire különböző utasításokat kell végrehajtani, tehát az utasítás nem emelhető ki, akkor az ON X után egy exkluzív utasítás-lista is állhat.

Pl. 10 ON X ' PRINT A : C = 3 ' PRINT D ' FOR J = 1  
TO 7 :U (J) = 0:NEXT  
20 GOSUB 100 ...

Ezt a hagyományos módon a következő programrészlet oldja meg:

10	ON X GOTO 1000, 1100, 1200
20	GOSUB 100...
1000	PRINT A : C = 2 : GOTO 20
1100	PRINT D : GOTO 20
1200	FOR J = 1 TO 7 : U (J) = 0 : NEXT :GOTO 20

Egyetlen kikötés, hogy az ON utáni utasítás -lista nem tartalmazhat újabb ON utasítást, vagy '-os IF-et. /Lásd később/.

Aposztrofhoz érve a program-végrehajtás abbamarad, és a következő sornál folytatódik. Az előző példában tehát a program minden X mellett a 20. sorban folytatódik. Mivel a programsor hossza nincs korlátozva, az ' bevezetésével az ON utasítás határtalan lehetőségeket nyit a programozó számára. Komplette eljárások írhatók meg egyetlen sorban.

#### IF

A relációkról elmondottak alapján nem meglepő, hogy az IF tulajdonképpen az ON kétértékű megfelelője. Hogy a BASIC kompatibilitása megmaradjon, a THEN szócskát megtartottuk, bár jelentése itt megegyezik az '. További ' segítségével az ELSE funkciója valósítható meg. /Ekkor a szigorúan vett ON-ban a reláció 0 értéke 2 kellene legyen. Ez az egyetlen különbség./

Az alábbiakban példák láthatók helyesen felírt IF utasításokra:

```
IF A < D THEN A=-A : B = A
IF A > D THEN A= Q ' B = P
IF A >= D PRINT 'A ' B
IF A > D GOTO '10'100
IF A >< D PRINT 'A:C = 5 'B:E = 6+B
IF A = D 'B = 0 : PRINT Z ' S = 4 : GOTO 100
```

#### PLOT X,Y

A CR változó által meghatározott színű pontot teszi ki a képernyő /x,y/ koordinátájú pontjára. A / 0,0 / pont a bal alsó sarokban van. X a vízszintes, y a függőleges koordináta. Az utasítás CR értékét nem változtatja meg. A koordinátákra nézve:  $0 \leq X \leq 127$  vagy 63;  $0 \leq Y \leq 95$

#### POKE I,J...

J értékét az I című memóriahelyre teszi. I-re ugyanaz érvényes, mint PEEK (I) nél. ezzel a különbséggel, hogy itt, ha  $I < 8192$ , akkor az adott címmel egy OUT /gépikódú/ utasítást hajt végre. /Output perifériák kezelhetők./ J után,-vel elválasztva további kifejezések is írhatók. Ezeket a soronkövetkező címekre írja be.



Ha I nagyobb mint 65536, akkor itt is a második lap címei érvényesek.

### POP

Ha FOR ciklusból, vagy BASIC ezubrutinból NEXT vagy RETURN utasítás nélkül akarunk kilépni, akkor ezt POP-utasítás után tehetjük meg. A ciklus és a ezubrutin eltárol egy visszatérési címet a stack memóriába. NEXT és RETURN utasításnál ezeket a gép mindig előveszi a stackból. Ha tehát többször kilépünk NEXT vagy RETURN nélkül, a stack lassanként megtelik. Az is előfordulhatna hogy egy félbehagyott ciklus keletkezik, és egy későbbi NEXT nem azt a ciklust zárja le, amit kellene. Ezt akadályozza meg a POP utasítás.

```
Pl: 10 FOR J = 1 TO V : IF A (J) = c (k) THEN POP:
      GOTO 100 ' NEXT
PRINT
```

Kinyomtató utasítás. A kulcsszó után a nyomtatási lista áll. Ez kifejezésekből, változókból, a CUR-ból, szám ill. stringkonstansekből állhat. A nyomtatási lista elemei között vessző és pontosvessző az elválasztójel. Ezek egyben a nyomtatási formátumot is meghatározzák. A pontosvessző a listaelemek egyszerű egymás után írását eredményezi. A vessző a következő zóna elejére állítja a kiíró. Ebben a gépben 1 sor 4 zónából áll, amelyek egyenként 8 vagy 16 karaktert tartalmaznak. Ha a listaelem után nem áll semmi, az a kiírás után egy soremelést eredményez.

Üres PRINT utasítás egy sort emel. A PRINT kulcsszót a kérdőjel teljes mértékben helyettesíti, elég tehát csak azt írni PRINT helyett.

PRINT #X;

Az OUTPUT perifériák kezelésére szolgáló utasítás. Az X szám vagy kifejezés, az Output-eszköz száma. #0 a TV-display-t jelenti, míg a #1 a PRINTER-t. A további output eszközök kezelését az INPUT-hoz hasonlóan szintén külön kell megírni. Ebben az OUTPUT# VEKTOR-ról írottak A CUR hatása nem jelenik meg a printeren, a CUR mindig a display-en érvényes.

## READ X1, X2

A DATA utasítás párja. READ hatására X1, X2 stb. változók rendre felveszik a DATA utasítások után felsorolt értékeket. Ezt úgy lehet szemléltetni, hogy az összes DATA utáni adatokat sorban betesszük egy táblázatba és a READ utasítással egyenként olvassuk ki őket. A READ-nak van egy "mutatója", amit minden értékváltozás után eggyel tovább állít. Újabb READ utasítás-onnan folytatja a táblázat olvasását, ahol az előzőt abbahagyta.

Ha a DATA után álló kifejezés hibás, akkor a hibaüzenet a READ sorában keletkezik.

Ha kevesebb az adat, mint a READ, akkor OD Error-t üzen.

## REM

A REM után megjegyzések helyezhetők el. Végrehajtáskor a gép ezeket a szövegeket egyszerűen átugorja.

## RESTORE

A DATA-val definiált adatmező elejére állítja a READ "olvasó-mutatóját". Ha a RESTORE után egy kifejezés áll, akkor azt kiszámolja, és a READ "olvasó-mutatóját" ennek a sornak az első DATA-jára állítja. Ha nincs DATA a sorban, akkor az ezután következő első DATA-ra áll.

## RETURN

Visszatérés szubrutinból, GOSUB-utasítás párja. GOSUB nélküli RETURN esetén P<sub>p</sub> Error keletkezik.

## RUN

Elindítja a programot. Mielőtt elkezdene végrehajtani a programot, először törli a változótáblát, és csak utána tér rá a legkisebb számú sorra. Ha RUN után egy szám áll, akkor a végrehajtás az adott számnál kezdődik. Hatása parancs módban sem azonos a GOTO-val, mert a GOTO nem csinál változó törlést. Pl. egy programhiba miatt leállt program tovább futtatható egy másik sortól a GOTO segítségével. RUN esetén elvesznének a változók értékei.



## SAVE "NÉV"

Program eltárolása kazettás magnóra. Idézőjelek között a rekord nevét kell beírni. CR megnyomása előtt a magnót kell elindítani felvétel állásban. Tárolás alatt a gép fűtyül. A tárolás \$4016-tól a Basic Text végéig tart.

## (LET )

Ilyen utasítás nincs ebben a BASIC-ben, az értékadás egyszerűen  $A = x$  alakú, ahol A egy változó, x pedig egy kifejezés. A LET szó nem is szerepel a kulcsszavak között.

## ÜZENETEK - HIBAÜZENETEK

A gép bekapcsolásakor, ill. minden RESET alkalmával bejelentkezik. Ha a BASIC nem működőképes, akkor inicializálódik. /Pl. bekapcsolásakor, ill. olyan programhiba esetén, ami a rendezerváltozókat megváltoztatta/. Ekkor kiírja a szabad memóriaterületet, és alapállapotba hozza a BASIC-et. Ha RESET-kor a BASIC "életképes" volt, ez nem történik meg. Ekkor a program megmarad, és a gép csak OK üzenetet ír.

Minden végrehajtott parancs után OK üzenet keletkezik. BRK-val megszakított program esetén BREAK üzenetet ad, és kilistázza a sort, ahol a futás megszakadt.

A BASIC hibaüzenetek általános alakja:

- hibás parancs esetén: V ERROR,
- hibás program esetén: V ERROR, és a hibás sor listája.

V itt a hibára jellemző kód, amit alább részletesen ismertetünk.

Programhiba esetén a rossz sor listája is megjelenik, segítve ezzel a hiba javítását. /Ez a kiírt sor a cursorral azonnal kijavítható./

A HIBAKÓDOK A KÖVETKEZŐK:

- BS      BAD SUBSCRIPT : Méreten kívül eső tömbindex
- CN      CONTINUE ERROR: Nem folytatható a futás
- DD      DOUBLE DIMENSIONED VAR: Kétszer dimenzionált változó
- IQ      ILLEGAL QUANTITY: Az argumentum kívül esik az értelmezési tartományon
- OD      OUT OF DATA : Több a READ utasítás, mint a DATA
- OM      OUT OF MEMORY: Elfogyott az engedélyezett memóriaterület
- OV      OVERFLOW : Túlcsordulás, ábrázolhatatlanul nagy eredmény keletkezett
- PP      POP ERROR : NEXT FOR nélkül, RETURN GOSUB nélkül vagy POP FOR ill. GOSUB nélkül
- SL      STRING TOO LONG: 255-nél hosszabb STRING keletkezett.
- SN      SINTAX ERROR : Szintaktikus hiba
- TM      TYPE MISMATCH : Az előfordult kifejezés típusa nem megfelelő
- US      UNDEFINED STATEMENT : Nem létező sorra történt hivatkozás
- /Ø      DIVISION BY ZERO : Ø-val történt osztás

Speciális karakterek kódjai

5	BELL	/ harang	/	11	→	/cursor jobbra /
6	INS	/ INSERT	/	12	CLS	/képernyő törlés/
7	DEL	/ DELLETE	/	13	CR	/ CARRAGE RETURN/
8	↓	/cursor le	/	14	TAB	/ tabulátor /
9	↑	/cursor fel	/	15	HOME	/ Kép elejére /
10	←	/cursor balra	/			



Ezt a gépet elsősorban BASIC programok számára fejlesztettük ki, ezért assembler támogatása alapkiépítésben nincs /Bővítésnek megvehető/. A monitor csupán az alapvető funkciókat látja el: memóriaterületet listáz, memóriába adatot tölt, memóriaterületet kazettára kitárol és a programot a megadott címtől elindítja. A monitor BASIC-ből CALL B92 /decimális/ hívható be. A monitor új sort kezdve jelzi, hogy a felhasználó rendelkezésére áll. Ezután gépelhetők be az egyes parancsok. A monitorban minden szám hexadecimálisan értendő.

# MONITOR PARANCSONK

D X Y vagy #D X Y

Memóriaterületet listáz X címtől Y címig. Hogyha a végcím (Y) hiányzik, akkor a D hatására -a képmérettől függően- 128 vagy 256 bájt íródik ki, soronként 8 vagy 16 bájt. A sorok elején a kettőspont után a sor első bájtjának címe van. Ha D után nem áll szám, onnan folytatódik a kiírás, ahol legutóbb abbamaradt. A Ø kezdőcímű listázás nem lehetséges, ekkor a gép nem veszi figyelembe az X címet.

Ha a D előtt # áll, akkor az előbbi szabályoknak megfelelően a printerre listáz.

:X P Q R ...

Memóriaterületet tölt be. X az a cím, ahová az első számot / P / teszi. A Q R ... stb. számok egymás után a soronkövetkező címekre kerülnek be. A CR megnyomásakor mindig kiírja a következő címet és új adatot vár. Üres sorra visszatér a monitorba.

A cím és az első adat között egy betűköz áll. További betűköz már nem szükséges, a számok sorban, kétjegyenként töltődnek be. Idegen /nem space vagy hex szám/ karakterre a betöltés abbamarad. Mivel D a lista elé kettőspontot ír, ezért az általa listázott sorok cursor-al javíthatók. CR-re a javított sor töltődik be.

G X

Programot indít el az X címtől.

A program végén, ha csak valami különös ok nincs, a monitorba célszerű visszatérni egy RET utasítással.

S X Y Z

Kitárolja kazettás magnetofonra az X-től Y-ig terjedő memóriaterületet. A rekord neve Z lesz. A név a második címet követő első nem space karaktertől a sor végéig tart. Kitárolás közben a beépített hangszóróból a magnetofonra felvett hang hallható. A monitorból kitárolt rekord formátuma azonos a BASIC-ével, ezért a beolvasás mindig BASIC-ben történik.

Az itt felsoroltaktól eltérő parancs-betűkre a gép hibát üzen.



RESET

A készülék bekapcsolása után BASIC-ben ébred. Előtte memória tesztet végez és beállítja a BASIC rendszerváltozót. Ezt a folyamatot inicializálásnak nevezzük. Ha a gép valamilyen okból végtelen gépi kódú ciklusba kerül, innen a RESET-gombbal téríthetjük vissza. Ez az NMI felhasználásával azoftver közbeiktatásával ugrik a 0 címre. Ez a RESET újra készíti a memória tesztet, de e közben a memóriatartalom nem változik, tehát programot nem töröl. A teszt után megnézi, hogy a BASIC "életképes"-e még. Ha minden rendben van, beugrik a BASIC-be és folytatódhat a programozás.

Ha valamilyen okból a rendszerváltozók átíródtak, a BASIC újra inicializálódik. Ekkor úgy jelentkezik be, mint bekapcsoláskor, és így a BASIC program is elveszik. Persze a gépi kódú rész még így is megmaradhat, hiszen azt a BASIC inicializálódás nem érinti.

A RESET kapcsán van még egy említésre méltó dolog. Nevezetesen az, hogy a BASIC-be ugrás előtt a gép megnézi, hogy a 2000 /Hex/ címen 0 van-e. Ha az van, nem megy a BASIC-be, hanem a 2001 címre ugrik. Ezáltal mód van a software-rendszer teljes módosítására, ROM-ban tárolt programok közvetlen behívására.

Mivel a gépben van hely ROM-bővítésre, az előbb említett tulajdonság minden külső csatlakozó vagy bővítés nélkül felhasználható.

KAZETTÁS ADATRÖSZÍTÉS /HANGKELTÉS/

Mivel mind az írást, mind az olvasást azoftverrel oldottuk meg, lehetőség van olyan programok /saver és loader/ készítésére, amelyek más gép számára készítenek kazettát, ill. más gépek kazettáit tudják elolvasni.

Ez adott esetben hasznos lehet, de közvetlen programcsere sajnos nem lehet így végezni. Legalábbis BASIC szinten nem! Mindenesetre kis ügyességgel adatok, szövegek vagy gépkódú programok így átvihetők egyik gépről a másikra.

Ahhoz, hogy ilyen saver-loader programokat írjon valaki, már alaposan tisztában kell lennie mindkét gép "lelkivilágával". Ezt elősegítendő, most leírjuk a gép tárolórendszerét.

Először lássuk a rekord formátumát!

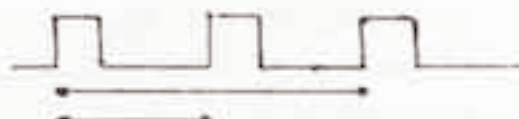
Minden rekord egy kb. 2 másodperces headerrel kezdődik. Ez a szinkronizálást szolgálja. A header csupa LO bit-et tartalmaz. A headert egy promt-byte /A5/ zárja. Ez után a rekord neve jön, amit egy NULL karakter / 00 / zár. Ezt az eltárolt terület kezdőcíme és hossza követi. Ez után jönnek az adatok és végül egy ellenőrző bájt. Ez a bájt az adatok összegének legkisebb helyiértékű bájtja. A rekord legvégén jön egy blockzáró karakter. Ha ez nem nulla, akkor megint jön egy block, amelynek headerje rövidebb, és nincs neve, de egyébként az előzővel azonos szerkezetű. Ha 0 a záró karakter, akkor befejeződik a rekord.

header	A5 prom	név	00	SL start LO	SH HI	LL length LO	LH HI	data	CH check byte	BE block end
--------	------------	-----	----	-------------------	----------	--------------------	----------	------	---------------------	--------------------

A kiadott adat bájtok között nincs szünet, sem paritás-bit, a rekord tehát egy bit-folyam. A magnetofonra kitárolt jelalak a következő.

" 0 "

" 1 "



Tehát egy bit akkor 1, ha két szinkronizáló jel között középen van egy harmadik impulzus is. A magnetofon ezt a jelalakot nem tökéletesen viszi át, de a jelölt idők, a nyávogástól eltekintve állandók.



Ilyen -tárolásra alkalmas- jeleket hangbit föl-le kapcsolgatásával /a keyboardmező 00 és 08 címének megszólításával/ lehet előállítani. Ez egyébként vonatkozik mindenfajta hangkeltésre is.

Ezzel a megoldással tehát tetszőleges soros formátum előállítható, csak pontosan ki kell számolni a program futási időit.

A beolvasás szintén elég egyszerűen történik. Ha a keyboardmező negyedik címét olvassa a 0. biten a soros bemenetet kapja. Nincs más dolga, mint megszámolni, milyen hosszú ideig van itt LO ill. HI. Ebből már egy egyszerű összehasonlítással eldönthető, hogy a bit HI vagy LO. Más formátumú kazetták ugyanezen az elven olvashatók, csak legfőbb nem ilyen egyszerű a dekódolásuk.

A bit formátum megváltoztatásán kívül gondoskodni kell még arról is, hogy a rekord formátuma is megegyezzen, ehhez azonban alaposan kell ismerni a kérdéses gépet. A HOMELAB 3. gépkazettás rendszere egyébként teljesen azonos a korábbi HOMELAB /AIRCOMP 16/ géppel.

#### INPUT - OUTPUT RUTINOK

- RST 1    A magnokezelés használja
- RST 6    Szabad a felhasználó számára. Ezek a restartok
- RST 7    egy-egy vektor által mutatott címre ugranak.
- RST 3    Az RST-vektorok helye rendre        4043 403F .  
Input egy karakter. A 4002 vektoron keresztül működik. Az A-ba beveszi az éppen lenyomott gomb ASCII kódját. Ha semmi nincs lenyomva, akkor A=0. Ha F1 vagy F2 is le van nyomva, az A-ban akkor is csak a gomb kódja lesz, de a gép elugrik a 4006 vektor által mutatott címre. Hogy F1 vagy F2 volt lenyomva, az a D regiszter 1. bitjéből derül ki. Alap esetben a 4006 vektor olyan programra mutat, ami a grafikus karaktereket aktiválja.

- RST 4    Általános pointer rutin.

$$A \leftarrow (DE) \quad \text{és} \quad DE \leftarrow (DE) + 1$$

- RST 5      Output rutin. Az A-ban levő karaktert kiírja a display-re vagy a printerre. c4.  
A 0-tól F-ig a kontrol-karaktereket adja, 10-től 1F-ig pedig a kontrol-karakterek képét, vagyis a karaktergenerátorban 0-tól F-ig levőket. Így tehát a valódi 10-től 1F-ig lévő karaktereket nem lehet ezzel kiírni.
- 0404      Az outputot a printerre állítja.
- 0423      Az outputot a display-re állítja.  
Figyelem! A BASIC utasításai mindig visszaállítják az outputot a display-re, tehát ez az átállítás csak gépikódból marad hatásos.
- 0546      Egy sort bevezető rutin.  
Inputot vár, míg egy CR-t meg nem nyomtak. Akkor beveszi az adott sor első 63 karakterét a 4060-nál kezdődő inputbufferbe. Ide mutat a DE pointer, és ezt kell használni a kiolvasáskor is.
- RST 2      Az 546 rutin meghívása után ez a rutin az input-buffer tartalmát adja át a programnak.  
E rutin minden meghívásakor az A-ba kerül az input szöveg következő karaktere.  
Az utolsó karakter átadásakor a 2 flag 1 lesz.



- 037C A MONITOR kezdőcíme.
- 01A0 Kiírja a DE tartalmát hexadecimálisan.
- 01A5 Kiírja a A tartalmát hexadecimálisan.
- 03F0 Az A-ba bevesz egy ASCII kódokkal leírt hex bájtot. A kódsorozatra a DE regiszterpár mutat. Mindig az első két karaktert értékeli ki. Ha nem hex szám karaktert talál, befejezi a számbavételt.  
A DE pointert a kiértékeléssel együtt tolja előre.
- 01DC A HL-be bevesz egy ASCII kódokkal leírt hex számot. Az előbb leírtak érvényesen itt is, kivéve hogy itt a B regiszter mondja meg, hogy hány karaktert kell figyelembe venni a bevételkor. Ha B nagyobb mint 4, akkor csak az utolsó négyet értékeli ki. Ha B=0, akkor végig kiértékel, tehát a karaktersorozat utolsó négy hex szám karakterét veszi be.
- 0188 Táblázatból szöveget ír ki.  
A szövegekben csak az első 128 karakter szerepelhet. A legmagasabb bit az egymást követő szövegek elválasztására szolgál. A szöveg utolsó karakteréhez tehát hozzá kell adni 80 -at. A táblázatot egy FF karakterrel kell kezdeni, és a HL regiszterpárnak erre a címre kell mutatni. Ezután jönnek a szövegek, az előbb leírt módon elválasztva. A B regiszter mutatja meg, hogy a táblázat hanyadik szövegét írja ki.  
A C regiszterbe 0-t kell betölteni!
- 03E9 DE pointerrel megkeresi az első nem space karaktert az input szövegben. A DE erre a karakterre fog mutatni. Figyelem! Ezt a rutint csak az input bufferben szabad használni.

- 0533 Átkapcsolja a memóriát a II. lapra.  
 Figyelem! Az input , output és hangkeltő rutinok ezt automatikusan visszakapcsolják!  
 A stack átkapcsoláskor a 402B-be kerül és csak 7 mélységben használható.
- 05BE Visszakapcsolja a memóriát az I. lapra.
- 0108 Hangkeltés. A H regiszterben a hang hossza, az L regiszterben pedig a hang magassága van. Ez a rutin nem végzi el az átkapcsolást.
- 00F6 A képszinkronhoz szinkronizálja a programot. Addig vár, míg egy TV-kép kiszerkesztése be nem fejeződik. Ha csak e rutin meghívása után fordulunk a VideoRam-hoz, akkor elkerülhető a display-n különben megjelenő "szemetelés".  
 Ez a rutin sem vált memórialapot, tehát meghívása előtt át kell kapcsolni. /természetesen csak akkor, ha egyáltalán van II. lap a rendszerben./
- 061F Load Header
- 0616 Load egy bájt az A-ba
- 074E Load az első érvényes rekordot
- 0751 Load egy rekordot.  
 A névre a DE pointer mutat, és a nevet 00 22 vagy 60 karakter zárja
- 061C Save header
- 06C9 Save egy bájt az A-ból
- 07A3 Save egy rekordot. A BC-ben a kezdőcím, a HL-ben a végcím van. A névre u.a. érvényes, mint Loadnál.

#### BASIC RUTINOK

- 1D4D A BASIC kezdőcíme.
- 18E1 Hangkeltés. Az A tartalmazza a hang magasságát, a C pedig a hosszát.

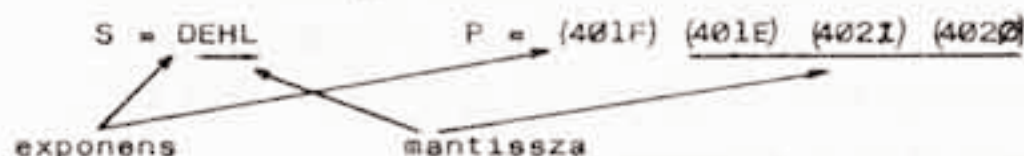


- 1BE7 Dallamjáteszés. /BEEP /  
A DE pointer a dallamtábla elejére mutat. A dallamot a 20 karakter zárja le. A táblázatban lévő karakterek 21-től 3F-ig a ritmust, 40 fölött pedig a hangmagasságot határozzák meg.
- 1538 Kiír egy tetszőleges szöveget. Ebben minden karakter szerepelhet. HL mutat a string elejére, E pedig a string hosszát adja.
- 1DE0 Plot gépikódból.  
Az A-ban az Y, az L-ben az X koordináta van. A szint a 404F rendszerváltozó határozza meg. Egyébként a Basic-kel azonos módon működik.

### ARITMETIKAI RUTINOK

Az aritmetikai 2 akkumulátort használ; az egyik a P akkumulátor /401Etől 4021-ig terjedő 4 bájtt/, a másik az S ami az E D L H regiszterekből áll.

Az akkumulátorok alakja a következő:



Az aritmetika működésének szabályai a Basic-nek megfelelően értendők.

Az általánosan felhasználható aritmetikai rutinok a következők: /a művelet a S és az P között értendő, az eredmény a P-be kerül/.

071B	+	0713	-
07CB	*	06A9	/
0C49	^	0A23	AND
0A18	OR	0C63	>
0C60	<	0C72	><
0C68	=	0C7C	=>
0C77	=<	1098	>=<

A függvények ugyan olyanok mint Basic-ben.

Az argumentum mindkét akkumulátorban szükséges. Az eredmény a P-ben keletkezik.

0974	INT	0994	ABS
0998	SGN	09AA	RND
0B68	COS	0B2C	SIN
0B78	TAN	0AA8	ATN
0C3C	SQR	0BA4	EXP
0BEB	LOG		

Egyéb hasznos aritmetikai rutinok:

063A	$P \leftarrow HL$
064D	$P \leftarrow NORM(S) : S\text{-et } P\text{-be normalizálja}$
09E6	$HL \leftarrow INT(S)$
0A09	$HL \leftarrow INT(P) \text{ es } DE \leftarrow INT(S)$
0E86	$P \leftarrow -P$
0BBC	$P \leftarrow NOTP$

#### RENDSZERVÁLTOZÓK

A két bájtos változóknál az alacsonyabb címen található a LO-bájt a következőn pedig a HI-bájt.

4000-01 BASIC, USR vector.

Az USR függvény kezdőcíme: 4000 a LO bájt  
4001 a HI bájt

Kezdőérték: 157C

4002-03 INPUT Vector.

Az ide beírt cím lesz az egy karaktert beevő rutin kezdőcíme.

Kezdőérték: 035C

4004-05 OUTPUT Vector.

Az ide beírt cím lesz az egy karakter kiíró rutin kezdőcíme.

Kezdőérték: 0283



4006-07

F1/F2 VECTOR.

Ha egy gomb mellé az F1 vagy az F2 is le van nyomva, e szerint a vector szerint ugrik el az input rutin folytatni a dekódolást. A kezdőértéknek megadott rutin a grafikus jeleket adja.

Amikor ide ér az input rutin, a A-ban a beütött gomb kódja van, és a Z ill. C status-flag-ek mondják meg, hogy az F1 vagy az F2 volt lenyomva.

Kezdőérték: 0600

4008

SHIFT ALTER flag.

Ez mutatja meg, hogy Shift nélkül nagy vagy kisbetű legyen. Ha páros kisbetű van.

4009

SCROLL stop flag.

Ha ez a flag 0, akkor a képalján megáll a kiírás és csak a SHIFT/Space hatására megy tovább.

400A

32/64 betű flag.

Ha 1 akkor 64 betűsakiírás, ha 2 akkor 32 betűs, de ehhez hardver átkapcsolás is kell.

400B

Buffer-pointer

Ez az output buffer pointere, szövegkiíráshoz használják.

400C-0D

RST 1 VECTOR

RST 1 hívásakor e szerint ugrik tovább.

400E-0F

INPUT \* Vector.

INPUT \* 1, ill. ennél nagyobb számú input-perifériák lekezelésére.

402E-2F	Current Line	cli.
	Az aktuális, éppen végrehajtott BASIC sor száma.	
4030-31	Start of BASIC Text	
	<u>Kezdőérték:</u> 40A0	
4032	Flags for BASIC	
	BASIC flegek, a felhasználó számára értéktelen.	
4033-34	Auxiliary Stack pointer. A stack pointer átmeneti tárolására szolgál.	
4035-36	Statement pointer for CONT. A következő végrehajtható utasítás címe a RAM-ban.	
4037-38	Next item for READ. A következő érték címe a RAM-ban, amit a READ olvasni fog.	
	<u>Kezdőérték:</u> 40A0	
4039-3A	Current Line for CONT. Erre a sorra tér vissza CONT utasításnál.	
403B-3E	Random Number last	
	A legutóbbi 32 bites véletlen bitsorozat.	
403F-40	RST 7 VECTOR	
4041-42	Free	
4043-44	RST 6 VECTOR	
4045-46	Free	
4047-48	Display pointer for screen-editor	
	Display pointer az editor működéséhez.	
4049-4A	ERROR VECTOR	
	Ide kell beírni az errorrt kiszolgáló rutin címét. Alapesetben a normál hibaüzenetre mutat.	
	<u>Kezdőérték:</u> 1581	



404B	INTERRUPT FLAGS	c12
404C	Security flag Védi a programot, ha az értéke 00. <u>Kezdőérték:</u> 10	
404F	CR változó. Ez a BASIC CR változója.	
4052-53	Vector for Interpreting A BASIC-fordító rutin kezdőcíme. Bővítéskor kell, a felhasználónak értéktelen. <u>Kezdőérték:</u> 16B2	
4054-55	Vector for Listin A BASIC-listázó rutin kezdőcíme. Bővítéskor kell, a felhasználónak értéktelen. <u>Kezdőérték:</u> 0167	
4056-57	Vector for Error A BASIC-error üzeneteket generáló rutin kezdőcíme. Ezt csak bővítéskor kell módosítani, felhasználónak értéktelen. <u>Kezdőérték:</u> 1581	
4058-59	Start of token table A BASIC kulcsszókészletét tartalmazó táblázat kezdőcíme. <u>Kezdőértéke:</u> 1C3C	
405E	PRINTER POS. A printer fejének pozíciója.	
405F	PRINTER FLAG. A printeléskor használt flag.	
4060-9F	OUTPUT és INPUT BUFFER, a kiírás és beolvasás gyűjtője.	

404B	INTERRUPT FLAGS	c12
404C	Security flag Védi a programot, ha az értéke 00. <u>Kezdőérték:</u> 10	
404F	CR változó. Ez a BASIC CR változója.	
4052-53	Vector for Interpreting A BASIC-fordító rutin kezdőcíme. Bővítéskor kell, a felhasználónak értéktelen. <u>Kezdőérték:</u> 16B2	
4054-55	Vector for Listin A BASIC-listázó rutin kezdőcíme. Bővítéskor kell, a felhasználónak értéktelen. <u>Kezdőérték:</u> 0167	
4056-57	Vector for Error A BASIC-error üzeneteket generáló rutin kezdőcíme. Ezt csak bővítéskor kell módosítani, felhasználónak értéktelen. <u>Kezdőérték:</u> 1581	
4058-59	Start of token table A BASIC kulcsszókészletét tartalmazó táblázat kezdőcíme. <u>Kezdőértéke:</u> 1C3C	
405E	PRINTER POS. A printer fejének pozíciója.	
405F	PRINTER FLAG. A printeléskor használt flag.	
4060-9F	OUTPUT és INPUT BUFFER, a kiírás és beolvasás gyűjtője.	



4000	-	4001	USR vector
4002	-	4003	RST 3 input vector
4004	-	4005	RST 5 output vector
4006	-	4007	F1 / F2 vector
4008			SHIFT ALTER flag
4009			Stop Flag for scroll
400A			32/64 karakter flag.
400B			buffer-pointer
400C	-	400D	RST 1 vector
400E	-	400F	INPUT * vector
4010	-	4011	PRINT * vector
4012	-	4013	Monitor-pointer
4014	-	4015	Cursor-pointer
4016	-	4017	Highest Memory pointer
4018	-	4019	End of Basic text
401A	-	401B	End of variable table
401C	-	401D	End of string-space 1
401E	-	4021	Primary accumulator
4022	-	402B	Auxiliary stack
402C	-	402D	End of string-space 2
402E	-	402F	Current line
4030	-	4031	Stat of Basic Text
4032			Basic Flags
4033	-	4034	Auxiliary stack pointer
4035	-	4036	Statement pointer for CONT
4037	-	4038	READ-pointer
4039	-	403A	Current line for CONT
403B	-	403E	Last Random Number
403F	-	4040	RST 7 vector
4041	-	4042	Free
4043	-	4044	RST 6 vector
4045	-	4046	Free

A következő oldalon a HOMELAB III. személyi számítógéphez vásárolható bővítések használatának leírása található.

Ezek a következők:

Bővített BASIC	§ 2000
ASSEMBLER	§ 2800
Duplapontos Aritmetika	§ 3000

A félreértések elkerülése végett ezeket a bővítéseket külön kell megvásárolni a géphez.

### BŐVÍTETT BASIC

A Bővített Basic elnevezésű program egy 2K-s toldalék az alap Basic-hez. A szűkös 8K-t 10K-ra bővítve számos olyan új lehetőség valósulhatott meg, ami a használat során sokaknak hiányzott.

A Basic Bővítés használata rendkívül egyszerű. Csak be kell helyezni a beégetett IC-t /közvetlenül a Basic mögé/ és bekapcsoláskor automatikusan aktivizálódik a bővítés. Ennek első látható jele, hogy a gép magyar nyelvű hibaüzeneteket küld. /A Basic többi része megmarad angol nyelvűnek. Az utasítás-készlet magyarra fordítása kimaradt, mert ezt a legtöbben fölöslegesnek tartották/.

És most lássuk az új utasításokat, függvényeket!

### UTASÍTÁSOK

DELETE	EDIT	ELOAD
ESAVE	GOSUB#	GOTO#
KEY	MERGE	REPEAT
UNTIL	VERIFY	



A következő oldalon a HOMELAB III. személyi számítógéphez vásárolható bővítések használatának leírása található.

Ezek a következők:

Bővített BASIC	§ 2000
ASSEMBLER	§ 2800
Duplapontos Aritmetika	§ 3000

A félreértések elkerülése végett ezeket a bővítéseket külön kell megvásárolni a géphez.

### BŐVÍTETT BASIC

A Bővített Basic elnevezésű program egy 2K-s toldalék az alap Basic-hez. A szűkös 8K-t 10K-ra bővítve számos olyan új lehetőség valósulhatott meg, ami a használat során sokaknak hiányzott.

A Basic Bővítés használata rendkívül egyszerű. Csak be kell helyezni a beégetett IC-t /közvetlenül a Basic mögé/ és bekapcsoláskor automatikusan aktivizálódik a bővítés. Ennek első látható jele, hogy a gép magyar nyelvű hibaüzeneteket küld. /A Basic többi része megmarad angol nyelvűnek. Az utasítás-készlet magyarra fordítása kimaradt, mert ezt a legtöbben fölöslegesnek tartották/.

És most lássuk az új utasításokat, függvényeket!

### UTASÍTÁSOK

DELETE	EDIT	ELOAD
ESAVE	GOSUB#	GOTO#
KEY	MERGE	REPEAT
UNTIL	VERIFY	

DEC	HEX§	FORM§
FRA	FSW	MAX
MIN	MOD	ROUND
STRING§	VAR	

RENDSZERVÁLTOZÓ

PRG

SPECIÁLIS KARAKTEREK

%

§

Bevezettünk egy új rendszerváltozót, a PRG-t. Ez csak olvasható változó. Azt adja meg, hogy a lehetséges Basic lapok /lásd EDIT/ közül éppen melyikben vagyunk.

Van két új speciális karakter:

% a mögötte álló számkonstanst binárisan értelmezi.  
A % mögött ezért csak 0 vagy 1 állhat. A bináris számkonstans hossza legföljebb 16 jegy lehet.

Pl.:           A = %10011           /A=19/

§ a mögötte álló számkonstanst hexadecimálisan értelmezi.  
A § mögött ezért csak 0 ... 9 és A B C D E F állhat.  
A § után álló jelsorozatból a gép az utolsó 4 jegyet értelmezi.

Pl.:           A = §3F39           /A=16185/

UTASÍTÁSOK

## DELETE A-B

A kulcsszó után egy cimintervallum áll. A cimintervallum a LIST-tel azonos módon értendő.

A DELETE a cimintervallum által meghatározott programsorokat kitörli. Ha a DELETE után nem áll semmi, az összes sort törli.



A gépben egyszerre több -egymástól független- Basic program elhelyezése lehetsége, és ennek megfelelően parancsmód is több lehet. Ezeket Basic lapoknak nevezzük. Az EDIT N ezek között biztosít átmenetet.

Alapesetben a gép a 0. lapon van. Tegyük fel, hogy oda már írtunk egy programot. Ha most kiadjuk az EDIT 1 parancsot, a gép kitorlí a változótáblát és átlép egy másik lapra. Ott új, az előzőtől független program írható. /Tehát újra lehet használni ugyanazokat a sorokat/. Ha itt listázunk, vagy itt adunk RUN parancsot, akkor az csak ezen a lapon lévő programra lesz érvényes.

EDIT 2-vel újabb lapot nyithatunk, míg EDIT 0-val visszatérhetünk a 0 lapra /alaplapon/.

Tehát az EDIT N parancsmódban áttesszi a vezérlést az N. lapra. Ha már korábban megnyitottuk ezt a lapot, akkor nem töröl változótáblát. Ha ilyen számú lap még nem volt, akkor megnyitja a következő lapot, és töröl a változótáblát. /Figyelem! ha Pl. 5 lap van nyitva, és EDIT 8-at adunk, akkor új lapot nyit, de az 6. lesz/.

Megnyitott lapok megszüntetése nem lehetséges. Ha egy lap szükségtelen, egyszerűen csak törölni kell belőle a programot. Hogy éppen melyik lapon dolgozik a gép, azt a PRG rendszerváltozó mutatja meg. Az alaplapon a PRG=0.

A különböző lapokon elhelyezett programok közös változótáblával dolgoznak, de a programok egymástól függetlenek, és egyszerre csak az egyik lapon futhat program.

Az egyik lapon futó program átugorhat egy másik lapra, vagy meghívhat egy másik lapon lévő szubrutint. Ezekre külön utasítások szolgálnak.

Magnóra tároláskor az összes megnyitott lap eltárolódik függetlenül attól, hogy melyik lapon adtuk ki a SAVE parancsot. Ugyanigy a LOAD is beveszi az összes kitárolt lapot.

Ha az EDIT után nem áll szám, akkor az az EDIT 0-val azonos. A programban kiadott EDIT N megállítja a programot és a gép parancsmódba megy az N. lapon.

GOTO # N,A

Ez az utasítás az N. lapon lévő program A. sorára ugrik. N is és A is lehet kifejezés az alap-Basic GOTO-jánál megismert szabályok szerint.

Pl.: GOTO PRG +2,1000 relatívan két lapváltás és ott GOTO 1000.

GOSUB # N,A

Ez az utasítás az N. lapon lévő program A. soránál lévő szubrutint hívja. N és A ugyanaz, mint a GOTO-nál.

A szubrutin végén, a RETURN utasítás hatására a futás visszatér az eredeti lap megfelelő sorába.

ELOAD " NÉV "

ESAVE " NÉV "

ELOAD X,"NÉV "

ESAVE X," NÉV "

Az első formának azonos a hatása a korábbi SAVE ill. LOAD utasítással.

A második forma különböző perifériákra történő tárolást ill. olvasást jelent. /X=1 a magnót adja/.

Ezekkel az utasításokkal kell majd például a Bar-Code programokat beolvasni, vagy egy lényegesen gyorsabb magnós tárolást megvalósítani.



VERIFY " NÉV "  
VERIFY X," NÉV "

d5

Ezzel az utasítással a kazettára felvett programokat lehet ellenőrizni. /Gépi kódú felvételt is!/.

A VERIFY a LOAD-hoz hasonlóan működik. "Elolvassa" a kazettát, de nem tölti be a memóriába, csak összehasonlítja azzal. Ha nem talál eltérést, OK üzenettel tér vissza. Ha volt hiba, "Értelmetlen" hibaüzenetet ad.

X ugyanaz mint ELOAD-nál

A VERIFY után a név hiányozhat, akkor az éppen következő programot ellenőrzi.

MERGE " NÉV "  
MERGE X," NÉV "

Új lapot nyit és a kazettán eltárolt programot betölti erre a lapra.

EXT

Alap-Basic-ben ez az utasítás HEX 2000-re ugrott. Mivel a bővítés erre a helyre került, ez az utasítás a Bővített Basic-ben HEX 2800-ra ugrik.

Alapkonfigurációban ide az ASSEMBLER kerül.

KEY

A KEY utasítás megváltoztatja a funkciós gombok jelentését. KEY utasítás után a betűk és az F1/F2 gombok kombinációjával a Basic legfontosabb kulcsszavai gombnyomásra bevíhetők. Újabb KEY utasítás visszaállítja az eredeti állapotot, ahol a funkciós gombokkal a grafikus karakterek érhetők el.

REPEAT : ..... : UNTIL R

d6

Ez a FOR - NEXT-hez hasonló ciklusszervező utasításpár. A REPEAT és UNTIL közötti programrészletet /ciklusmag/ addig ismétli, amíg az UNTIL után megadott R reláció értéke hamis.

```
P1.:      1Ø A=Ø
          2Ø REPEAT : PRINT A : A=A+1 : UNTIL A=1Ø
```

ez a programrészlet ekvivalens a következővel:

```
1Ø A=Ø
2Ø PRINT A : A=A+1 : IF A 1Ø THEN GOTO 2Ø
```

/Ez a példa csak formailag mutatja a REPEAT - UNTIL működését. Valódi haszna itt nem látszik/.

## FÜGGVÉNYEK

DEC /A\$/

Ha A\$ tartalma egy HEX szám ASCII-képe, akkor DEC megadja ennek decimális értékét. /A HEX szám kettes kódomens kódban értendő/.

Az AS elején lévő betűközöket a DEC figyelmen kívül hagyja. Ha a HEX szám 4 karakternél hosszabb, csak az utolsó 4 jegyet veszi figyelembe.

```
P1.: DEC (" EFE5") = -4123
      DEC ("F") = 15
```

HEX\$ /X/

Ez a függvény előállít egy 4 karakteres stringet, ami az X kifejezés értékének hexadecimális ábrázolású képe lesz

$$-2^{15} < x < 2^{16}$$



P1.:  $\text{HEX}\$ (4123) = "101B"$        $\text{HEX}\$ (123) = "007B"$       d7  
 $\text{HEX}\$ (-4123) = "EFE5"$  de       $\text{HEX}\$ (61413) = "EFE5"$

$\text{MIN} (11, 12, \dots, \text{In})$

$\text{MAX} (11, 12, \dots, \text{In})$

Ezek a függvények az argumentumban felsolt kifejezések értékei közül a legkisebbet ill. a legnagyobbat adják.

P1.  $\text{MIN} (4, 7, 3, 9) = 3$

$\text{FSW} (X, Y, Z)$

$\text{FSW} (X, Y, Z, U)$

Az első forma X függvényében Y vagy Z kifejezés értékét adja.

Ha  $X = 0$  akkor az FSW értéke Y lesz

Ha  $X \neq 0$  akkor az FSW értéke Z lesz

a második formában ugyancsak X függvényében ad értéket:

Ha  $X = 0$  akkor az FSW értéke U lesz

Ha  $X \neq 0$  akkor az FSW értéke Y lesz

Ha  $X \neq 0$  akkor az FSW értéke Z lesz

Példa: a következő program az A I tömb legnagyobb elemét keresi meg

$T = 0$  : FOR I = 0 TO 100

$T = \text{FSW} (A (T) > A (I), I, T)$

T-ben a legnagyobb értékű elem indexe lesz.

$\text{MOD} (X, Y)$

$X, Y \neq 0$  tetszőleges számok. Ekkor

$\text{MOD} (X, Y) = X - \text{ABS} (Y) * \text{INT} (X / \text{ABS} (Y))$

vagyis X "maradékát" adja Y-al osztva.

P1.:  $\text{MOD} (26, 3) = 2$

X,Y lehetnek törtek is. Pl. a következő program szöveget számol át radiánból fok, perccé a MOD segítségével.

d8

```
10 INPUT " radián :"; R
20 V = V*180/P;
30 ? INT (R) ; " fok ";
40 ? INT (MOD (R*60 , 60)); " perc"
```

FRA (X)

X tört részét adja.  $FRA(X) = X - INT(X)$

Pl.:  $FRA(PI) = .141593$   
 $FRA(-PI) = .858407$

ROUND (X,Y)

Az X kifejezés értékét az Y. tizedesre kerekíti.

Pl.:  $ROUND(PI,+4) = 3.1416$   
 $ROUND(1532,-2) = 1500$

VAR (V)

V tetszőleges típusú változó azonosítója.

A VAR függvény megadja, hogy a V változó értéke hol van eltárolva a memóriában.

Ha V még nem kapott értéket, akkor a VAR értéke 0 lesz. REAL típusú változó esetén a VAR által adott címen kezdődő 4 bájt a változó értéke. Ez az Alap-Basic részben leírt formában értendő. String változónál csak 3 bájt értékes. Az első a string hosszát adja, a másik kettő pedig azt mutatja meg, hogy hol kezdődik a string a memóriában.

STRING\$ (X)

STRING\$ (X,Y)

Ez a függvény egy X hosszúságú stringet ad, ami csupa Y kódú karakterből áll.

Ha Y hiányzik, akkor az X hosszúságú string a betűköz karakterből fog állni.



A Z kifejezés értékének formatált képét adja meg.  
 X az egészrész, Y pedig a törtrész számjegyeinek számát adja meg. Az egészrészt és a törtrészt a tizedesjel választja el, és egy további karakter a szám előtt az előjel. Így a FORMZ egy  $X + Y + 2$  hosszúságú stringet ad.

Ha a Z értéke már nem fér el ebben az ábrázolási tartományban, akkor az előjel függvényében csupa + vagy csupa - string keletkezik.

Figyelem! A FORMZ nem kerekít !

A FORMZ szerkezetét az X és az Y paramétereken kívül egy további paraméter is szabályozza. Ez a HEX 4042 címen lévő rendszerváltozó, ami 6 bitjében a következőket jelenti.

X	x	5	4	3	2	1	0	§ 4042
---	---	---	---	---	---	---	---	--------

bit

- 0 Ha 0, akkor a + előjelet nem írja ki /helyette betűköz áll/.  
 Ha 1, akkor a + előjelet is kiírja.
- 1 Ha 0, akkor tizedespontot ír.  
 Ha 1, akkor tizedesvesszőt ír.
- 2 Ha 0, akkor az értéktelen nullákat nem írja ki a törtrészben.
- 3 Ha 1, akkor 0 törtrész esetén kiírja a tizedesjelet, és egy nullát.  
 Ha 0, akkor sem tizedesjelet, sem nullát nem ír.  
 A 2. bit prioritása nagyobb.

4 Ha 0, akkor a nulla egészrészt nem írja ki d10  
/csak egy tizedesjelet/.

Ha 1, akkor egy nullát ír a tizedesjel elé.

5 Ha 0, akkor az előjelet a bal szélső helyre írja.

Ha 1, akkor az előjelet közvetlenül az egészrész  
elé írja.

Példa:

POKE\$4042, 00001011 esetén

FORM\$(3,3,5) = "+ 5,0 "

POKE\$4042, 00110001 esetén

FORM\$(3,3,1/2) = " +0.5 "

#### HIBAÜZENETEK

BS Rossz tömbindex!

CN Nem folytatható!

DD Ujra méretezett tömb!

IO Nem értelmezhető szám!

OD Több READ, mint DATA!

OM Túl kicsi a tár!

OV Túl nagy szám!

PP Stack hiba!

SL Túl hosszú szöveg!

SN Értelmetlen!

TM Rossz adattípus!

US Hivatkozás nemlétező sorra!

/0 0-val való osztás!



gomb	F <sub>1</sub>	F <sub>2</sub>
Q	LIST	SIN
W	LOAD	COS
E	MERGE	SQR
R	REPEAT	INT
T	UNTIL	MIN
Y	RETURN	MOD
U	READ	VAL
I	INPUT	INKEY
O	BEEP	CHR <del>g</del>
P	POKE	PEEK
D	REM	MID <del>g</del>
Ó	END	STRING <del>g</del>
A	RUN	TAN
S	SAVE	ATN
D	VERIFY	RND
F	FOR	ABS
G	NEXT	MAX
H	GOSUB	ROUND
J	RESTORE	LEN
K	PLOT	POINT
L	CUR	STR <del>g</del>
U	CALL	LFT <del>g</del>
Z	DELETE	EXP
X	EDIT	LOG
C	CONT	USR
V	STEP	SGN
B	POP	FSW
N	GOTO	FRA
M	DATA	ASC
Á	DIM	FORM <del>g</del>
É	EXT	RGH <del>g</del>

A bővített monitor és assembler alapvető fontosságú programcsomag azok számára, akik a Basic programozás mellett gépkódban is hozzá akarnak férni gépükhöz. Tekintettel arra, hogy a gépkódú programozás kb. 10-1000-szeres sebességnövekedést jelent a Basic-hez képest, számtalan esetben elengedhetetlen, hogy a teljes programot, vagy annak kritikus részeit, gépkódú programmal valósítsák meg.

A második szempont, hogy gépkódban alakítható ki a feladathoz illeszkedő optimális adatstruktúra is, ami szintén a sebesség növekedését, de még inkább a felhasznált memória csökkenését eredményezi.

A harmadik terület, ahol gépkódot kell alkalmazni a perifériák és egyéb külső eszközök szoftverje. Itt szintén a sebességi előny dominál, illetve az, hogy az Interrupt-kérések "azonnal" lekezelhetők, ellentétben a Basic-kel, ahol ki kell várni az aktuális Basic utasítás végét, ami adott esetben több ms is lehet.

Végül meg kell említeni, hogy a rendszerprogramok nyelve is a gépkód, a gép szoftver lehetőségeinek kibővítésére is ez az egyetlen út kínálkozik.

Tehát mindazok, akik elérték a Basic korlátait, és tovább szeretnének lépni, feltétlenül a gépkód felé kell, hogy forduljanak.

A gépkódú programozás rejtelmeivel itt természetesen nem foglalkozhatunk, erre vonatkozóan jónéhány szakkönyv eligazít. Ugyanigy nem térhetünk ki a Z80 mikroprocesszor utasításkészletének ismertetésére sem. Ezeket mint ismertnek tételezzük fel, tehát aki nem tud valamit, itt álljon meg és nézze át a szakirodalmat.



A Bővített Monitor és Assembler /röviden BMA/ 2 K területet foglal el. Megvásárolható ROM-rezidens formában. A ROM-rezidens forma értelemszerűen gépbeépítve működik. A BMA-t a Basic-ből lehet hívni EXT paranccsal.

Ugyancsak érvényes a BMA-ra a Basic MON parancsa is. A BMA bejelentkezése után a gép új sort kezd. Ez, és a villogó cursor jelenti, hogy a gép parancsra vár. A parancsok itt egybetűsek és ezek után áll az adott parancshoz szükséges további szám, illetve szöveg.

A gép minden parancsot a CR megnyomására hajt végre. A Screen-editor itt is működik. A Basic-kel azonos módon itt is bármi bárhol átírható a képernyőn és a CR hatására az aktuális logikai sor értelmeződik./Aktuális az a szövegsor, amelyikbe a cursor éppen van/.

Előljáróban talán még csak annyit, hogy a BMA-ban minden szám hexadecimálisan értendő. Ez 16-os számrendszer ahol a 10, 11, 12, 13, 14, 15 jeleket rendre az A, B, C, D, E, F betűk reprezentálják.

#### MOST LÁSSUK A PARANCsOKAT:

- D • HEX. formában kiír egy memóriaterületet a képernyőre.
- : Memóriatartalmat lehet beírni Hex. formában.
- S Memóriaterületet lehet kitárolni kazettára.
- G Programot indít el.
- F Egy adott területet feltölt egy megadott karakterrel.
- M Egy memóriaterületet átmásol egy megadott helyre.
- C Két memóriaterületet összehasonlít.
- T Assembly formában kiír egy memóriaterületet.  
/visszafordító/
- : Gépikódú programot lehet beírni Assembly formában.  
/oda fordító/
- R visszatérés a főprogramba.

Ha olyan parancs előtt # áll, melynek van ontputja, e3  
akkor az a printerre fog vonatkozni.  
Ilyenek a D, a C, a T parancsok.

E rövid áttekintés után lássuk az egyes funkciókat  
részletesebben is, X, Y és Z hex számokat fog jelenteni.

#### F X Y Z

X kezdőcímtől Y-ig feltölti a memóriát/még Y-t is/Z-vel.  
Z tehát egy kétjegyű szám. Ha ennél többet írunk, csak  
az utolsó két jegyet értelmezi és tölti be.

#### M X Y Z

X-től Y-ig /Y-t is!/ terjedő részt átmásolja Z helyre.  
Átmásolás után Z tartalma ugyanaz lesz, mint ami az X-é,  
Z + 1-é, min X + 1-é, stb.

Z-re nincs semmilyen megkötés, bárhova át lehet rakni.  
/A BMA "megnézi" Z viszonyát X-hez ill. Y-hoz és ennek  
megfelelően felülről lefelé vagy alulról fölfelé halad-  
va másol. Ez biztosítja, hogy másolás közben az eredeti  
tartalom nem vessz el/.

#### C X Y Z    vagy    #C X Y Z

Az X-től Y-ig terjedő területet /Y-t is!/ összehasonlítja  
a Z-nél kezdődő ugyanilyen hosszú memóriaterülettel. Az  
eltérő tartalmú címeket felsorolja tartalmukkal együtt.  
Az első /négyjegyű/szám címet jelent, a második az azon  
található adatot. A harmadik az a szám, ami a Z-től kez-  
dődő területben a megfelelő helyen áll. A különbségek  
felsorolását a BMA zónánként és automatikusan végzi el.  
Az összehasonlítás a jelzett terület végén, vagy az X-  
gomb megnyomásakor fejeződik be.  
A különbségek kiírását természetesen a Shift/space-val  
is meg lehet állítani, ugyanúgy, mint a Basic-ben.



E parancs hatására egy RET utasítás hajtódik végre, tehát a BMA visszatér az őt szubrutinként meghívó programba. Mivel a BASIC-ből is szubrutinként hívjuk a BMA-t, ezért a R parancs alkalmas a BASIC-be való belépésre, feltéve persze, hogy még megvan a visszatérési cím.

Ellenkező esetben G Ø vagy G 1D4D paranccsal, vagy egyszerűen csak RESET-tel lehet visszatérni.

T X Y vagy # T X Y

X-től Y-ig terjedő részt Z-80 mnemónikus kódba fordítja és kiírja. X-re és Y-ra pontosan ugyanaz vonatkozik, mint a D paranccsnál.

T a lista elé pontosvesszőt ír, utána jön az utasítás címe, majd a mnemónikus kód.

Ez a mnemónikus kód néhány kivételtől eltekintve a Zilog jelölésével azonos. A kivételek a következők:

Zilog standard			HOMELAB BMA	
ADC	HL, regiszterpár		ADC	regiszterpár
ADC	regiszter, szám		ADC	regiszter, szám
ADD A, /HL/			ADD	/HL/
SBC	/lX+d/, /lY+d/		SBC	/lX+d/, /lY+d/
BIT Ø, regiszter			BITØ	regiszter
BIT 1, regiszter			BIT1	regiszter
• • •			•	•
• • •			•	•
SET 7, regiszter			SET7	regiszter
CALL C, cím			CALLC	cím
• • •			•	•
• • •			•	•
CALL Z, cím			CALLZ	cím

feltételes JP, JR és RET utasításoknál ugyanigy.

IM	Ø	IMØ
IM	1	IM1
IM	2	IM2
IN	A, /n/	IN /n/
IN	reg /c/	IN regiszter

OUT-nál ugyanigy

OTIR	OUTIR
OTDR	OUTDR
RETN	RETR

A két byte-ot mozgató, vagy abszolút címre hivatkozó LD utasítások helyett MV használatos. /Az LD a szimpla regiszterek közti átvitelre van korlátozva.

#### ; X MNEMONIC

Ez a parancs a kettősponthoz hasonlóan működik. X címre berakja annak az utasításnak a kódját és operandusait, amit utána mnemónikus formában beírtak.

A mnemónikus kódokkal kapcsolatban ugyanaz érvényes, mint T-nél.

A relatív ugrásoknál /JR/ be lehet írni relatív címet, vagy azt az abszolút címet, ahova az ugrás szükséges. Ha címnek FF-nél kisebb szám áll a relatív, ha ennél nagyobb, akkor az abszolút cím lesz.

Formailag a visszafordító úgy működik, hogy a beírt sort a CR megnyomása után letörli és visszairja a Standard formátumot. Ez kettős célt szolgál:

Mint majd látni fogjuk, a felhasználónak nem kell mindig mindent precízen beírni az assembly formából, így az esetleges beírási mód helyett mindig egy tiszta, áttekinthető formátum látszik. Másrészt azonnal meg lehet győződni arról, hogy a gép valóban azt az utasítást vette be, amit szeretnénk volna.



Ha a gép nem tudta értelmezni a beírt szöveget nem történik meg a felülírás. A Cursor a helyén marad és újra kell írni az utasítást.

Minden sor bevétele után a BMA pontosvesszőt tesz, és kiírja a soron következő címet. Ide újabb utasítást lehet beírni, vagy üres sorral alaphelyzetbe visszatérni.

A D-: párhoz hasonlóan módosítani lehet a T-által kiírt listát. A cursorral visszalépkedve bármi -cím is- átjavítható, CR-re újra tölthető.

Figyelem! Ha a javítás során a kezdő-címek megváltoznak ügyelni kell, hogy az abszolút és relatív ugrások helyesek maradjanak. A gép nem végzi el a címek módosítását!

Mint már említettük a felhasználónak nem kell beírni azt a teljes formátumot, amit a gép kiír.

Első könnyítés, hogy egy komplett utasítás után bármi állhat, az nem zavarja a visszafordítást. Átjavított soroknál tehát szükségtelen letörölni a sor hátralévő részét, azt a gép már figyelmen kívül hagyja.

Ugyanigy nem kell ügyelni az utasítás operandusai közötti betűközökre, vagy a beírt utasítás helyére a soron belül.

Második, hogy a számot jelölő \$-jel elmaradhat, ha egyértelmű, hogy az adott helyen csak szám állhat.

/pl. ugrások/. Ahol regiszter is állhat ott ki kell tenni a \$-t, mert a HEX számok jegyei egyben regisztereket is jelölnek. A végzőrójel helyett lehet betűközt írni, a kezdő zárójel pedig csak akkor szükséges, ha az egyértelműség azt megkívánja.

A HOMELAB BASIC számábrázolási pontossága 6 számjegy. Ez sok esetben kevésnek bizonyul. Ennél lényegesen többre van szükség statisztikai, illetve adatfeldolgozási feladatoknál és olyan matematikai eljárásoknál, ahol a hibák összegződnek. Ezért kifejlesztettünk egy olyan aritmetikai bővitést, amely a számkijelzés pontosságát 14 számjegyre növeli.

#### A SZÁMÁBRÁZOLÁSRÓL

A duplapontos csomagban minden szám 8 byte-os lebegőpontos alakban van eltárolva. Ez egyben azt is jelenti, hogy a belső alapl műveletek relatív pontossága  $2E-17$ , míg az ábrázolható számok tartománya  $1E-32$ -től  $1E+32$ -ig terjed. Függvények esetén ez a relatív hiba valamivel nagyobb, de mindenképpen  $1E-15$  alatt marad.

A gép a számokat 14 jegyre kerekítve írja ki 0.1-től 99999999999999-ig törtalakban, azonkívül pedig normálalakban.

A szám, illetve a tizes kitevő + előjele elhagyható /kiíraskor a gép is betűközt ír a helyére/, míg a negatív előjelet mindig ki kell tenni.

#### A VÁLTOZÓK

A duplapontos bővítésben természetesen érvényben maradnak a korábbi változótípusok.

Az új, dupla pontosságú /ezután rövidítve D/ változó jele a felkiáltójel  $!$ . D változó neve egy betűből és a felkiáltójelből áll. Kétbetűs név itt nem használható.

D-tömbök azonosítására szintén egy betű használható, ami után a felkiáltójel és a zárójeles indexkifejezés áll. Vektor és mátrix azonosítója nem lehet azonos betű, a tömbök indexkifejezésében nem szerepelhet D-változó. Az index maximális értéke 255. Így összesen 31 D-változó és 31 D-tömb szerepelhet egy programban.



Az egyszerű D-változókat az első hivatkozás által már a gép automatikusan definiálja. Ekkor mind a 26 lehetséges változónak rezervál területet, tehát lefoglal 214 byte-ot a memóriából.

A D-tömböket mindig definiálni kell, nincsen automatikus definiálás a 10 alatti tömbökre sem. A definiálást a DIM utasításnál ismertetjük. Itt említjük még meg, hogy egy  $n \times m$  felső indexhatárú D-tömb definiálására  $6+8 \times n + 1/x / m + 1/$  byte-ot használ fel a gép.

Példák:

A! B! ... Z! A!/5/ B!/X,Y/ Q!/5,6/

### KIFEJEZÉSEK

Duplapontos /D/ kifejezés az, ami duplapontos utasításokban szerepel /lásd később/.

D-kifejezésekben használhatók a zárójelek, az összes aritmetikai művelet és reláció.

A logikai műveletek /AND OR NOT/ D-kifejezésekhez nem használhatók.

D-kifejezésekben természetesen szerepelhetnek D és real változók és konstansok is.

A függvények közül a következők állhatnak D-kifejezésekben:

ABS	ATN	COS	EXP
INT	LOG	SIN	SGN
SQR	TAN	VAL	

Nem használhatók az RND, PEEK, FRE függvények! /Ha mégis ezek értékére volna szükség egy D-kifejezésben, akkor előbb egy real változónak kell értéket adni ezekkel a függvényekkel és utána ezt a változót kell beírni a D-kifejezésbe/.

Ha a D-kifejezésben real típusú érték szerepel, akkor azt a gép D-vé alakítja és azon végzi el a műveleteket és függvényeket. Így számolás közben is mindig D-típusú részeredmény keletkezik.

A D-kifejezések további tulajdonságai azonosak az alap-BASIC-ben megismert real kifejezésekével.

### DUPLAPONTOS UTASÍTÁSOK

Egy utasítás akkor duplapontos, ha előtte felkiáltójel áll. Ez azt jelenti a gép számára, hogy ebben az utasításban minden számolást dupla pontossággal kell elvégezni. Az összes real típusú szám és változó D-típusúvá konvertálódik, és így megy végbe a kiértékelés.

Három D-utasítás lehetséges, nevezetesen

=/értékadás/            !DIM            !PRINT

### ÉRTÉKADÁS

Alakja az alap-BASIC-en túl háromféle lehet:

a/ !A! = D-kifejezés            értékadás D változónak.

A D-kifejezés kiértékelése közben minden real típusú változó vagy Constans D típusúvá konvertálódik, és ezzel az értékkel folytatódik a számolás.

b/ !A = D-kifejezés            értékadás real változónak.

D-kifejezés itt a számolások legvégén real-lé konvertálódik, és ezt az értéket kapja a real változó.

c/ !A! = VAL/A\$/            D-értékadás stringből.

Mivel duplapontosságú INPUT nincs, ez a módja D-értékek bevitelének.

INPUT "Szög:"; A\$/ : !A! = VAL/A\$



d/ !A\$ = STR\$/D-kifejezés/      String értékadás.      f4

A\$ a D-kifejezés értékének stringképét kapja.  
Az STR az egyetlen string fv. amelyben elő-  
fordulhat D-változó, ezért az egyenlőség után  
kötelező a STR\$, és a D-kifejezés után már csak  
a bezárójel állhat. /További stringműveletek nem!/

e/ A PI változó      a      D-értékét adja.

Mivel az értékadások különböző fajtaival az eset-  
leges Real      D, illetve D      real konverziók  
megoldhatók, külön konvertáló utasítások nincse-  
nek.

Real	D	D	real
!A!	= B	!A	= B!

## DIMENSIONÁLÁS

Alakja      !DIM A! /6,2/,      B!/5/,      C!/Q/...

A DIM előtt felkiáltójel áll, utána pedig a változók  
listája vesszővel elválasztva. !DIM után csak D-töm-  
böt lehet definiálni, real és D nem keverhető.

A D-tömbök indexkifejezésében real változók használ-  
hatók /D változó nem!/

Minden D-tömböt definiálni kell, a 10 alatti maximá-  
lis indexűeket is!

A duplapontos változók:

1 betűsek plusz felkiáltójel  
PI a D-értéke

Duplapontos műveletek:

+ - \* /

Duplapontos függvények:

ABS	ATN	COS	EXP	INT	LOG	SIN	SGN
SIN	TAN	VAL					

Duplapontos utasítások:

= /értékadás/ !PRINT !DIM

ezek előtt felkiáltójel áll, ha utánuk bárhol az utasításban van legalább egy felkiáltójel.